OFF-THE-SHELF AND FREE SOFTWARE TECHNOLOGIES FOR SPACECRAFT CONTROL & COMMAND : AN EXAMPLE, BALLOON-BORNE STABILISED GONDOLAS

André LAURENS
CNES - The French Space Agency
18, avenue Edouard Belin
31401 Toulouse Cedex 9 – FRANCE

Résumé

Les ballons sont des véhicules spatiaux permettant, à faible coût et sous des délais réduits, d'emporter des expériences scientifiques ou technologiques qui nécessitent de voler hors atmosphère ou de réaliser des mesures *in situ*. En cela, ils sont complémentaires des systèmes satellitaires.

Les nacelles pointées du CNES sont des plateformes génériques qui emportent sous ballon des instruments scientifiques, provenant essentiellement des communautés de l'aéronomie et de l'astrophysique, qui requièrent des services de stabilisation et de pointage analogues aux systèmes de contrôle d'attitude des satellites.

Dans le but d'accroître la flexibilité des nacelles et de leur permettre de s'adapter plus facilement à de nouvelles missions, les technologies mises en avant pour contrôle – commande sont celles des calculateurs industriels, des réseaux informatiques sol, du logiciel libre et, avant tout du langage Ada, car ce sont des solutions ouvertes, standard, puissantes, peu coûteuses et pérennes

Après avoir décrit brièvement le contrôle – commande des nacelles pointées et leur domaine d'application, cet article présente les différentes technologies et les grands principes de conception proposés pour atteindre les objectifs système.

Puis il se focalise sur les options d'architecture bord (applications Ada95 temps réel réparties), et s'attache à décrire le travail de prototypage et les développements préliminaires qui ont été conduit pour s'assurer de leur faisabilité. Ensuite est analysée l'applicabilité de ces solutions pour réaliser des applications globales de contrôle – commande, réparties entre bord et sol en s'appuyant sur des moyens de télémesure et télécommande fondé sur le protocole IP, tel que ceux que le CNES développe pour systèmes ballon.

En guide de conclusion, ce papier montre l'adoption des technologies ci-dessus pour d'autres programmes spatiaux, comme des plateformes de satellites et leurs charges utiles, peut induire de profonds changements dans la conception, les coûts, les durées et l'organisation des développements, de même qu'elle est à même d'ouvrir la porte à de nouvelles modalités de communication pour opérer les systèmes spatiaux.

Abstract

Balloons are low-cost, short development time space vehicles for science missions and technology in-flight experiments that need out-of-atmosphere or in-situ measurements, thus being complementary to the satellite.

CNES' stabilised gondolas are versatile space platforms used to fly science instruments – mainly for aeronomy and astrophysics – that need stabilisation and pointing capabilities, analogous to satellite attitude control subsystems.

In order to increase gondola flexibility to new missions, promoted control & command technologies are those of industrial computers, ground networks, free software and, over all, Ada language, for they are open, standard, powerful, low-cost and long-lasting solutions.

After a brief description of domain-oriented characteristics of stabilised gondola control & command, this paper introduces the various technologies and main design principles proposed to meet system-level goals.

Then focus is put on on-board architectures (Ada95 real-time distributed applications), and describes the prototyping work and preliminary development done to ensure feasibility. The paper then discusses the applicability of such solutions to global, ground-to-board, distributed control & command applications, through an IP-based telemetry & telecommand link, such as the one under development in CNES for balloon systems.

As a conclusion, this paper shows how adoption of the above technologies for other space programs – such as satellite platforms and payloads – may change design, development costs, duration and organisation, as well as it may open new ways in ground-to-board communication and spacecraft operation.

1 BALLOONS, GONDOLAS, STABILISED GONDOLAS

Balloon systems (also called aerostatic systems) are long time known space vehicles, used to fly science instruments and technology in-flight experiments.

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **13 JUL 2005**	2. REPORT TYPE **N/A**	3. DATES COVERED **-**
4. TITLE AND SUBTITLE **Off-The-Shelf And Free Software Technologies For Spacecraft Control & Command : An Example, Balloon-Borne Stabilised Gondolas**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **CNES - The French Space Agency 18, avenue Edouard Belin 31401 Toulouse Cedex 9 FRANCE**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release, distribution unlimited**		
13. SUPPLEMENTARY NOTES **See also ADM001791, Potentially Disruptive Technologies and Their Impact in Space Programs Held in Marseille, France on 4-6 July 2005., The original document contains color images.**		
14. ABSTRACT		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**UU**	**16**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

Science missions concern two main scientific domains : ***aeronomy*** – atmosphere chemical / thermal / dynamical study by way of spectrometry or sampling techniques – and ***astronomy*** (mainly astrophysics) by way of out-of-atmosphere measurements in infrared, submillimetric, UV, X or gamma wavelengths.

Technology experiments take advantage of out-of-atmosphere flights (e.g. solar cell calibration) or use balloon-borne gondolas as a space test platform to ensure feasibility prior to satellite operational usage.

Different kinds of balloons (infrared montgolfiers, pressurised stratospheric balloons, open stratospheric balloons, etc.) are available to achieve various kinds of missions, depending on payload mass, flight altitude (and acceptable altitude variation), season, latitude. Flight duration span from few hours to few weeks, and gondolas weight from few kilograms to some hundreds of kilograms (up to 2-3 tons).

Balloons are cheap space vehicles, and gondola solutions are traditionally based on extensive usage of off-the-shelf technologies. Thus balloon systems offer low-cost, short development time, multiple flights to payloads, allowing inter-flight evolution, and then being complementary to satellites.

Following the example of satellite systems, balloon systems can be broken down into three main subsystems : *a launcher*, the balloon itself plus flight train, parachutes and flight control equipments; *a platform*, the gondola; *a payload*, the science instrument.

Stabilised gondolas are among the biggest balloon-borne gondolas (some hundreds of kilograms, including instrument). They are versatile platforms designed to fly science instruments that need stabilisation and pointing capabilities, analogous to satellite attitude control subsystems. Since they fly various kinds of experiment, stabilised gondolas have to meet various requirements, along with low-cost, short delay constraints. So they are designed as generic platforms, in order to fit new missions and instrument evolutions.

As a keystone for genericity, stabilised gondola control & command shall use flexible architectures, with respect to mission requirements (flexibility of functions) and equipments (flexibility of interfaces). Cost / delay objectives are met by way of generic solutions and modern developments tools.

2 CONTROL & COMMAND TECHNOLOGIES FOR STABILISED GONDOLAS

In order to increase flexibility, new hardware and software solutions have been searched for all C&C functions : on board data handling / monitoring / stabilisation and pointing software, ground segment. This quest has been driven by a few principles :

- global architecture and function repartition leading to increased board autonomy, simplification of ground segment and ground-to-board communication, ground network communication principles applied to C&C;
- development organisation based on distinction between gondola generic items and mission-specific ones, and on basic / reusable components, this being applied to functions, hardware and data;
- same technologies on board and on ground (i.e. on board usage of ground computer and software technologies), for they are standard, powerful, low cost, open, long lasting solutions, described hereafter.

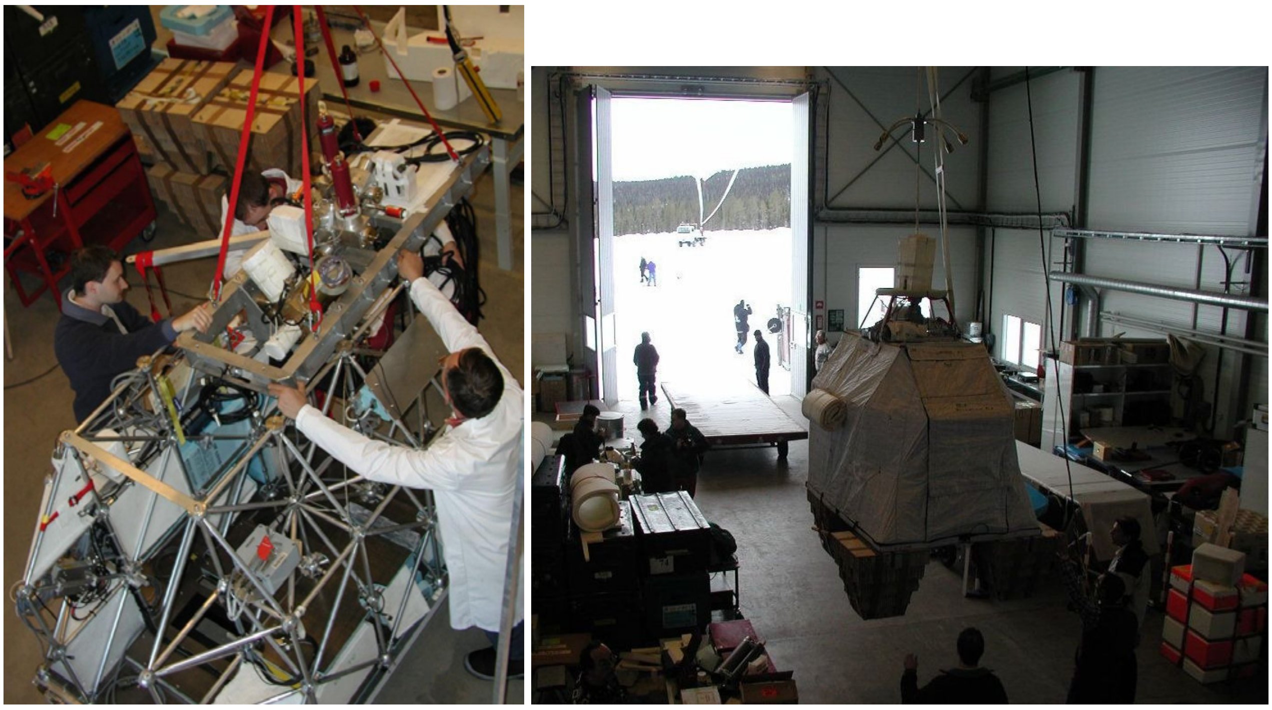

Figure 2-1 : An aeronomy stabilised gondola during integration (left) and ready for launch (right)

2.1 LINUX ON BOARD

Basic hardware architecture lies on industrial PCs, built from PC104 standard boards. They come now with powerful processors, various peripheral circuitry, and stay suitable to build on-board computers, with respect to usual mass and power consumption constraints.

A typical computer is made of :

- a full PC CPU board with 80x86 compatible processor, mouse, keyboard, screen and disk controllers, parallel, serial, USB ports, Ethernet controller, 16-128 Mb RAM, etc.;
- I/O boards : analog, digital, motor control, extra serial ports;
- a flash memory card used as a hard disk;
- external I/O modules are plugged to the PC through a daisy chain serial link, and used for housekeeping acquisitions and commands (analog or digital I/O, relay switch);
- several computers are interconnected through an Ethernet LAN with IP-based protocols.

Figure 2-2 : Examples of industrial PC components

All these features allow to run the on-board computers just like ground ones (this kind of hardware is used in ground industrial process control systems), thus with standard operating systems and development tools.

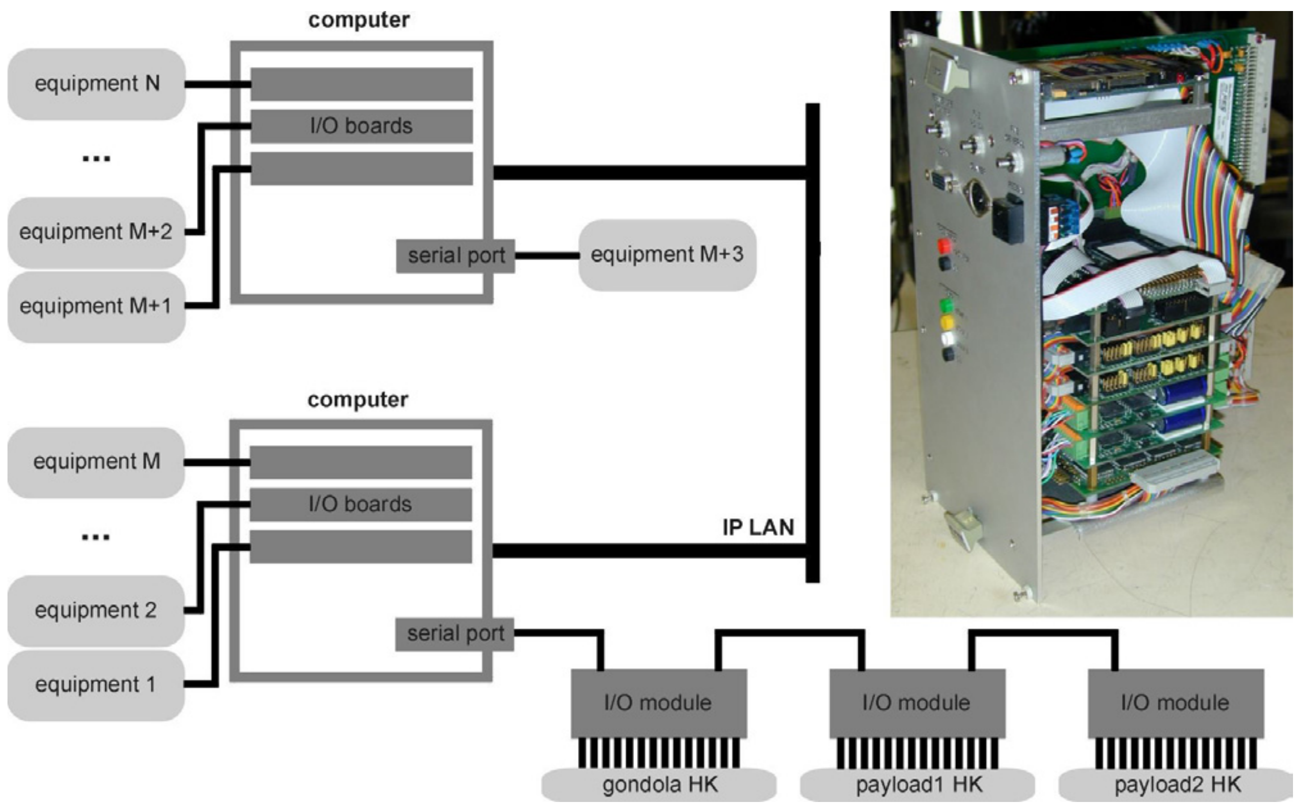

Figure 2-3 : A typical on-board computer architecture (left) and a PC104 stack (right)

Consequently, Linux appeared as the good candidate, since it can easily be run on industrial PCs, and a standard kernel can be used as soon as a compact distribution is chosen (see [LINUX]). Some PC104 board manufacturers already provide Linux distributions for their CPU boards. Almost all O.S. services are used : tasking, memory management, file system, standard drivers, network stack, etc. Only the display / window manager and graphic tools are left aside since no human operator is on board the gondola.

Such a platform makes a standard and convenient on-board computer, and provides the same level of services and development tools than ground computers. Using same processor and O.S. on board and on ground (our development and ground segment computers are desktop or laptop PCs under Linux), flight software executable files can be generated on ground computers and uploaded to the on-board computers through the network. **Thus, flight software development is no more crossed but native development.**

2.2 ADA AND FREE SOFTWARE

Ada has been selected as the core of software development environment, not only for it is the most powerful object-oriented general purpose programming language, with strong typing, structured features, generic units and exception handling (see [ADA ISO]). The specific reason is that it gives a standard and complete solution for real-time distributed embedded applications.

GNAT, the GNU Ada compiler, implements the whole Ada95 standard, and particularly the D and E annexes, respectively dealing with real-time and distributed systems (see [GNAT]). Annex D, along with built-in tasking features of the core language, are used to develop real-time embedded software. GLADE (GNAT's implementation of Annex E, see [GLADE]) allows to design a distributed application just as if it had to be run on a single computer, and to distribute it in a further step, by way of configuration directives, without the smallest change to source code.

Since no operational software development can be led without a convenient methodological frame, **HOOD** (see [HOOD]) has been chosen to be the design method of stabilised gondola C&C. Designed to be THE method for Ada software development, it combines hierarchical, object-oriented and real-time features. It is a true method, not only a formalism or a modelling language, and it is supported by at least one operational tool : STOOD (see [TNI]).

Along with GNAT, some Ada-related free software is used for ground segment development : *GtkAda and Glade-2* : a graphic toolkit and GUI builder based on Gtk+; *XmlAda* : an XML parser written in Ada; *AdaSockets* : an Ada interface to IP sockets (see respectively [GTKADA], [GLADE-2], [XMLADA], [ADASOCK]).

Last but not least, EAST data description language and associated tools are used for telemetry and telecommand packet description (see [EAST]).

2.3 NETWORK SOLUTIONS

Another keystone of stabilised gondola C&C development is the extensive usage of network technologies :

- on board : several computers on an Ethernet-IP LAN run a single distributed application (thanks to GNAT-GLADE);
- on ground : gondola control centre is made of a server application in charge of telemetry acquisition and storage and telecommand emission, and of processing & display clients that need not to be situated in the same place than the TM/TC station, but elsewhere on an IP network;
- system-wide : the IP-based next generation of CNES balloon telecommunication systems applies Internet-like communication to balloon systems (as well for integration activities and for the flight itself), and – why not? – will allow, combined with GNAT-GLADE, the development of ground / board distributed applications.

3 ON-BOARD ARCHITECTURE PRINCIPLES AT WORK

The above solutions had to be evaluated and tested in a realistic way, prior to make a decision of their usage for operational stabilised gondolas C&C. This has been done with the help of students from various French engineer schools and universities. Their work, described in [S-AC04], [S-ALDS03], [S-FP03], [S-KH04], [S-JPSV02] and [S-SS01], led to the conclusion that the promoted technologies were mature enough to allow operational developments and, moreover, that a gondola C&C based on such solutions proved feasible, reliable, flexible, and met the cost and delay constraints of balloon systems.

This chapter will now focus on the main topics of on-board architectures (real-time and distributed features) and how they have been tested by way of predevelopments and prototypes.

3.1 REAL-TIME APPLICATIONS WITH ADA AND LINUX

The first question to answer was : is the Ada – Linux couple able to support real-time applications such as ours?

That needs of course to characterize these applications. Studying their external requirements, they appear as “soft real-time” applications, that means :

- prominence of cyclic processes (such as attitude control loops) with frequencies ranging from few Hz to few tenths of Hz;
- tolerance on cycle durations and deadlines : around 1 ms;
- few non-periodic processes (such as telecommand processing, response time around 1s);
- higher priority for operational loops, lower for non-periodic processes.

One of our design principles leading to highly multi-task implementation, it becomes easy to match directly external requirements to processes (e.g. 1 control loop = 1 task), so that no extra real-time constraint comes from the underlying hardware / software architecture. That means that a given functional requirements (such as an attitude control loop running at 5 Hz, with a maximum delay of 100 ms between sensor acquisition and actuator command) may be directly translated, with the same accuracy, in a software application running on a standard computer + O.S. platform. On the opposite, in a traditional satellite application – sequencer-based software, domain-specific on-board bus – external requirements have to be broken down into low-level constraints (e.g. response time to a bus exchange request, time slot allocated to a piece of processing in the sequencer cycle) that are usually more severe.

On that basis, a multi-task breadboard has been set up to study the temporal behaviour of such applications in the Ada + Linux environment[1]. It comprises cyclic tasks, with scalable priority and frequency, built on the same model : run time is divided into :

- acquisition time, represented by a delay, since the task waits for input from a peripheral,
- computation time, represented by some statements simulating extensive CPU work,
- actuation time, represented by a delay, since the task waits for output completion,
- telemetry emission time, represented by some write-like statements, for an amount proportional to a TM packet size.

Some non-periodic tasks are added, with scalable priority and execution time, that are manually started.

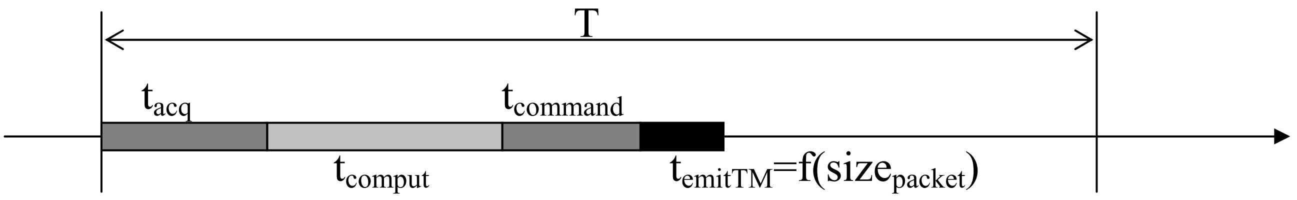

Figure 3-1 : Synopsis of a simulated cyclic task

Various application test cases have been generated (the different tasks of a case are obtained by Ada generic unit instantiation) and run, showing that cycle durations and deadlines were satisfied, in accordance with the above objectives. Benchmarks of typical heavy computation functions used in pointing systems (e.g. astronomical calculation) were also run and proved compatible with cycle durations and deadlines.

3.2 DISTRIBUTED FLIGHT SOFTWARE MOCK-UP

The next step of our evaluation process was then to test in quasi-full scale GNAT's distribution facilities. That has been done by developing a model of a distributed flight software.

3.2.1 How does the Ada95 Distributed Systems Annex work?

In order to understand the design principles of such an application, let us read the first verses of Ada95 Standard Annex E (see [ADA ISO]):

1. *This Annex defines facilities for supporting the implementation of distributed systems using multiple partitions working cooperatively as part of a single Ada program.*
2. *A distributed system is an interconnection of one or more processing nodes (a system resource that has both computational and storage capabilities), and zero or more storage*

[1] It has to be noticed that GNAT on Linux is an O.S.-based implementation, so that each Ada task is implemented by a Linux thread, and Ada task scheduling is delegated to Linux.

nodes (a system resource that has only storage capabilities, with the storage addressable by one or more processing nodes).

3. *A distributed program comprises one or more partitions that execute independently (except when they communicate) in a distributed system.*
4. *The process of mapping the partitions of a program to the nodes in a distributed system is called configuring the partitions of the program.*
5. *The implementation shall provide means for explicitly assigning library units to a partition and for the configuring and execution of a program consisting of multiple partitions on a distributed system; the means are implementation defined.*

All is said there. Provided the program units that make the application have been properly categorized[2] and the categorization hierarchy obeyed, then the "surprisingly simple" power of GLADE can be used :

- develop your application just as if it was to be run on a single computer,
- use GLADE configuration directives to define partitions and assign library units to them,
- compile and run;
- not a line of code has been written by the programmer for inter-machine communication, and changing something in the distribution configuration can be made without any change in the source code.

But how does it work? As shown on Figure 3-2, an non-distributed application made of 3 Ada units A, B and C, may be transformed into a 2-partition distributed application, Partition_X to which are assigned units A and B, and Partition_Y to which is assigned unit C. What are procedure calls between units in the non-distributed application stay procedure calls between units A and B since they are in the same partition. To achieve remote calls to unit C, GLADE will generate two extra units :

- a client stub for C in Partition_X, that has the same specification than unit C, and a stub body that transmits procedure calls through a network stream[3],
- a server stub for C in Partition_Y that listens for network requests, translates them into local calls to unit C, and sends back results.

The distributed application generated by GLADE then consists in 2 executable files (one per partition) that can be installed and run (even automatically) on the different computers of the final distributed system.

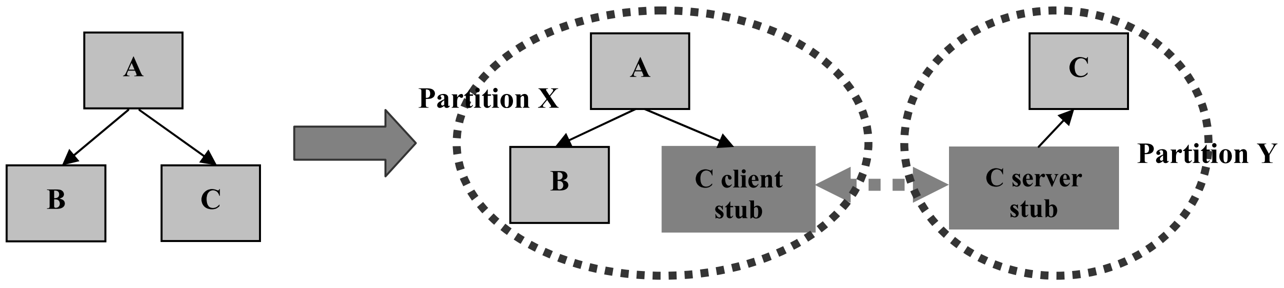

Figure 3-2 : GLADE's process for partition generation

[2] Categorization pragmas are defined for the library units, that restricts their declarations, child units, or semantic dependences : ***pure*** : no internal state; ***shared passive*** : defines only global data shared between active partitions; ***remote types*** : defines only types used in communications between active partitions; ***remote call interface*** : interface for remote calls between active partitions; normal (no categorization) : unrestricted.

[3] it makes usage of Ada streams, which is a core language feature introduced in Ada95 standard for this purpose.

3.2.2 Software architecture principles

The distributed flight software mock-up of course had to obey our autonomy rules (high abstraction level concepts, on-board guiding and mode control, on-board transfer functions and physical value computation) as well as our real-time design principles (mapping control loop on tasks, cyclic or not). In addition, the global software architecture presents a layer-oriented view of objects, seen as software translation or modelling of objects in the real world (see Figure 3-3).

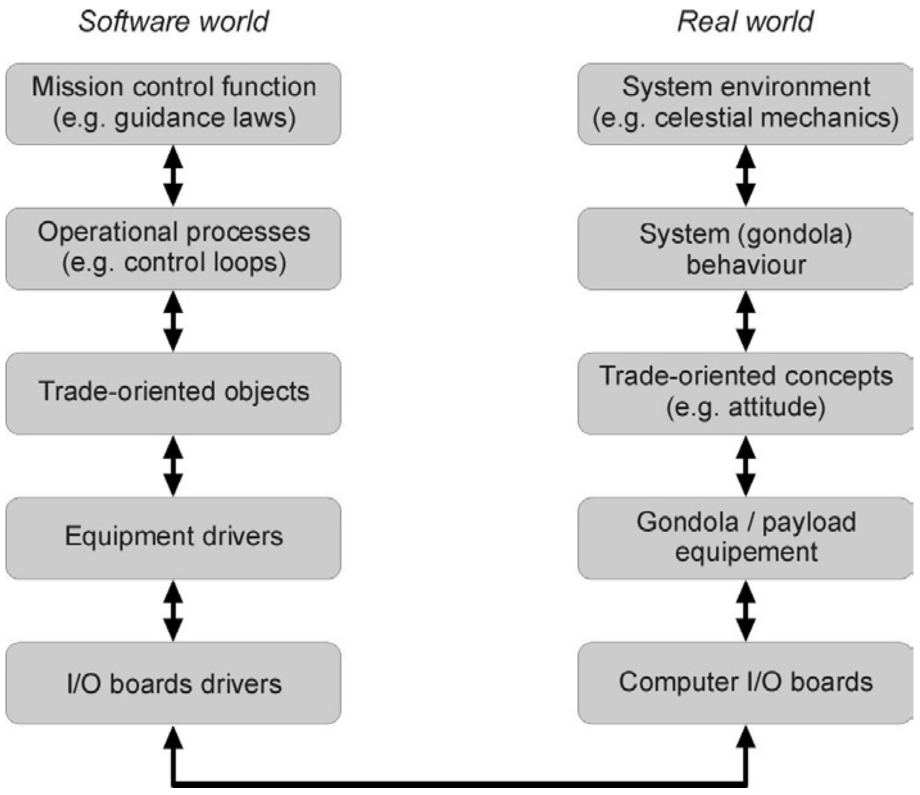
Software world
Real world
Mission control function (e.g. guidance laws)
Operational processes (e.g. control loops)
Trade-oriented objects
Equipment drivers
I/O boards drivers
System environment (e.g. celestial mechanics)
System (gondola) behaviour
Trade-oriented concepts (e.g. attitude)
Gondola / payload equipement
Computer I/O boards

Figure 3-3 : Layer-oriented view of software architecture

This view brings out two kinds of basic software components : I/O boards and equipment drivers.

Board drivers take advantage of Ada features : strong typing, encapsulation, readability, bit-level manipulation, O.S.-specific code encapsulation, optimized and highly efficient generated code. That makes 90%-portable drivers, with high level application programming interface.

Equipment drivers are designed to encapsulate equipment's computer interface (i.e. board address or id, channel or serial port number, message structure, communication protocols) as well as transfer function or time behaviour. They present a high level programming interface (e.g. "read angular speed" for a gyro) even independent of the precise equipment model, what prevents application-wide changes in case of equipment upgrade.

The above two kinds of drivers are of course candidate to become (and designed to be) reusable software components.

3.2.3 The flight software mock-up itself

To be significant for evaluation and feasibility demonstration purpose, the mock-up had to implement realistic flight software functions, in terms of kind of processing (real-time, computing, data exchange, etc.) and in terms of requirement coverage (housekeeping, pointing function, guiding, mission control, etc.). It had to run on a realistic hardware configuration : at least 2 computers, some real equipment. Last but not least, the distribution scheme should be as extensive as possible, on order to avoid toy cases and have a realistic vision of GLADE's capabilities – and possible weaknesses.

Hardware configuration :

- a couple of computers on an Ethernet IP LAN
- various kinds of I/O boards (analog, digital, serial, motor control)
- some real equipment (screw jack with home / end switches, GPS receiver) or placeholders (voltage source, LED, switch)

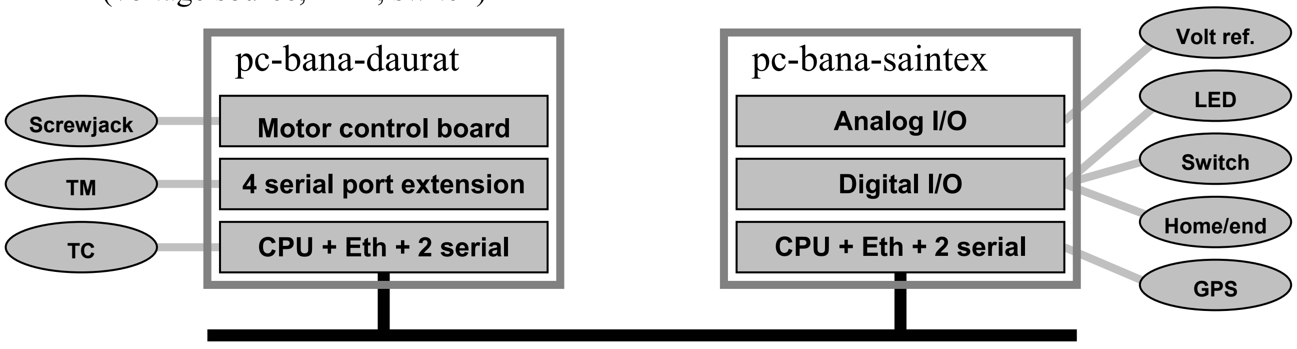

Figure 3-4 : Distributed mock-up hardware configuration

Functions :

Telemetry and telecommand management : channel interface supporting both serial (present telecom system) and IP sockets interface (next telecom system), data storage at each end, packet structure realistic definition, a single software object used in flight and ground software;

Housekeeping & monitoring : periodic acquisition of analog (temperature, voltage, current) and digital (relay status) values, emission in dedicated TM packets, on-demand relay switching;

On-board time management & localisation : acquisition of GPS time + loc message, on-board handling of GPS-synchronized time base, loc data server, software stopwatch service;

Guiding : target direction transform from inertial to altazimutal coordinate system, Sun tracking, astronomical calculations using GPS time + loc

Azimuth rough control : 10Hz control loop, simulated equipments (magnetometer, gyro, torque swivel, inertial wheel)

Elevation rough control : 1Hz open loop command of elevation screw jack (closed loop in the motor control board), complete equipment management (elevation target positions transform into screw jack commands, folding / unfolding procedure, home / end emergency actions);

Gondola supervisor : global control of flight software by way of a mode-related state / transition automate, authorisation / execution of telecommands.

In addition to the above mock-up specific code, some reusable components have been developed or refurbished : equipment drivers (GPS receiver, screw jack and home/end switches), I/O board drivers (serial, analog, digital, motor control), various general-purpose packages (types and operations for physical value computation, O.S. services encapsulation, astronomical and geomagnetic calculation).

Distribution :

The distribution process has been led by a certain number of obvious *a priori* criteria :

- minimisation of hardware-related coupling : board driver is located on the computer that owns the associated board, but not always the equipment drivers associated with equipment connected through the board;
- optimisation of coupling by data/control flow : highly coupled objects in the same partition, low-coupled objects may be in different partitions;
- distribution of high level processing with respect to CPU load concerns.

But how determine the good area for a partition? For extensive demonstration purpose, we chose to match each "big function" to a partition, and then assign the partitions to computers with respect to the above criteria.

Once built by GLADE process, the application could be installed (by a simple *ftp*) and run on the above hardware configuration. Reading the actual mock-up configuration file (see Figure 3-5), it can be noticed that :

- the configuration language is self-comprehensible, as well for library units assignment to partitions as for partition hosting onto computers, and one can figure out how easy become distribution configuration changes[4];
- you don't need to assign explicitly to partitions all needed library units: GLADE configuration tool follows Ada dependencies to find out the closure, and places them (i.e. copies or not) with respect to categorisation pragmas (a *pure* unit may be – and will be – copied wherever needed, a *remote call interface* unit shall be located where specified by configuration directives);
- the equipment driver for home / end switches (package *Butees*) is not assigned to the same partition than the board driver (package *Entrees_Sorties_Logiques*) for the digital I/O board to which the switches are connected; instead it is assigned to the same partition than the screw jack equipment driver. In addition to the fact that it does not prevent the software to run with the expected performance (time overheads introduced by remote calls on a private LAN are negligible compared to global software real-time requirements), it demonstrates the ability of GLADE technology to virtually abolish constraints traditionally put on software architecture by hardware topography.

Various tests have been performed, in both TM/TC configurations (serial and IP, with wire-to-wire connections[5]) and with a ground segment prototype using similar technologies than those of flight software. The on-board application proved working quite well and tests run showed good performance, according to system level real-time requirements. The relatively high number of partitions (compared to the minimum that could have been conceived, i.e. one per computer) had no visible impact on performance. That shows how pertinent and efficient is Ada distribution model, that allows to assign quite independently units to partitions and partitions to computers.

3.3 TOWARDS SYSTEM-WIDE DISTRIBUTION

For flight software mock-up test purposes, some man-machine interface was necessary, mainly to start on-board actions without setting up the whole telecommand data flow and associated processing, protocols and other paraphernalia. After first attempts with quite uncomfortable console-oriented I/O, it became obvious that windowed solutions could not be ignored, and this led us to make the first step towards system-wide distributed applications.

[4] for instance, moving a partition from a computer to another (i.e. changing one line of the configuration file and then recompiling the application) takes less time than moving a board from a computer to another.

[5] i.e. without real operational telecommunication systems : it is the usual procedure for gondola and payload tests, and that reinforces the advantages of standard interfaces between the various components of a space system and telecom equipment.

```
--------------------------------------------------------------------------------
Configuration logiciel_bord is

   pragma Starter(None);
   pragma Boot_server ("tcp","pc-bana-daurat.cst.cnes.fr:5557");

   p_main : Partition := (lv_nacelle_bord);

   p_controle_general : Partition := (lv_nacelle, controleur_nacelle);

   p_controle_azimuth : Partition := (controle_pointage_azimuth_primaire,
                                      magnetometre, gyro_z, roue_z, pivot);

   p_controle_elevation : Partition := (controle_pointage_elevation_primaire,
                                        inclinometre_y, verin, butees);

   p_controle_servitudes : Partition := (controleur_servitudes, servitudes,
                                         controle_thermique, controle_batterie,
                                         controle_on_off, acquisition_analogique,
                                         entrees_sorties_logiques);

   p_gps : Partition := (temps_loc, temps_loc.gps, temps_loc.base_de_temps,
                         datation_tm);

   p_tm_tc : Partition := (controle_tm_tc, voie_tm, voie_tc, mesures_bord,
                           type_acquitement_tc);

   p_guidage : partition := (guidage);

   procedure lv_nacelle_bord is in p_main;

   for p_main'Host use "pc-bana-daurat.cst.cnes.fr";
   for p_controle_general'Host use "pc-bana-daurat.cst.cnes.fr";
   for p_tm_tc'Host use "pc-bana-daurat.cst.cnes.fr";
   for p_guidage'Host use "pc-bana-saintex.cst.cnes.fr";
   for p_controle_azimuth'Host use "pc-bana-saintex.cst.cnes.fr";
   for p_controle_elevation'Host use "pc-bana-daurat.cst.cnes.fr";
   for p_controle_servitudes'Host use "pc-bana-saintex.cst.cnes.fr";
   for p_gps'Host use "pc-bana-saintex.cst.cnes.fr";

  for Partition'Storage_Dir use "bin";

end logiciel_bord;
--------------------------------------------------------------------------------
```

Figure 3-5 : Exemple of GLADE configuration file

The adopted solution consists in a distributed application, running on both on-board computer and ground development workstation, connected through an IP LAN, and using GNAT, GLADE and GtkAda technologies.

In our test tool, only man-machine interface is located on the ground computer, and all tested software runs on the on-board computers. But, as long as there is an IP LAN in between, nothing prevents to design a fully distributed application on a system-wide hardware configuration, made of several on-board and ground computers.

In the immediate future, these principles apply to test beds and instrumentation applications, comprising a PC104 stack connected to tested equipment, and running acquisition and low level data processing software, and graphical workstations running high level processing and display software[6].

Further, the next IP-based balloon telecom systems (described in [IP-JA00], [IP-CC00], [IP-JACC01]) that permit to extend and connect on-board and ground LANs through various physical

[6] note that on-board and ground computer need not be homogeneous : distribution on heterogeneous computers is fully supported by GLADE, including the usual endianness headache.

data links, using or not dedicated radio frequency channels, allow such a system-wide distributed application to run indifferently in the integration room or with the gondola in flight.

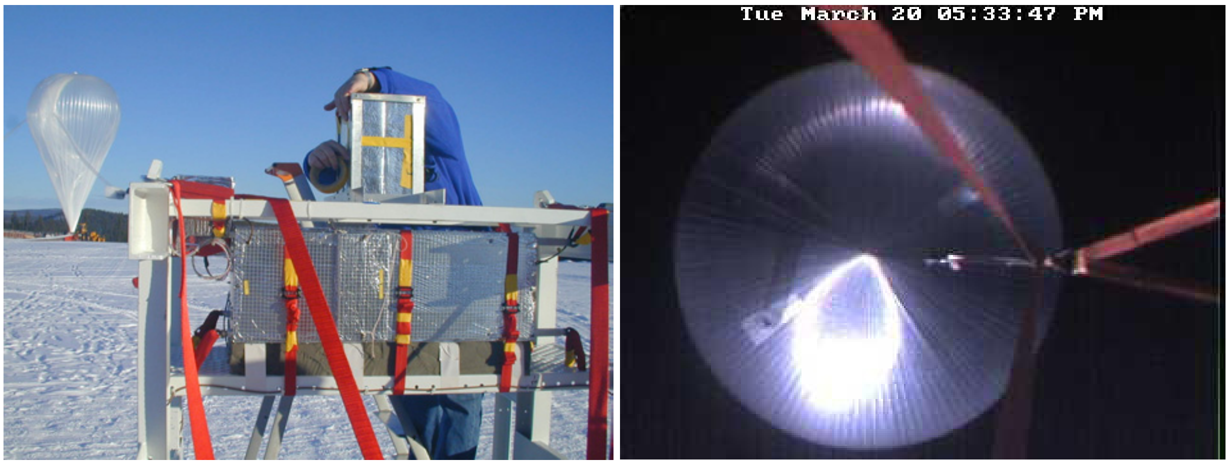

Figure 3-6 : First flight of IP router prototype – test gondola a few minutes before launch (left), balloon at float, seen with an on-board webcam (right)

4 DISRUPTION PERSPECTIVES FOR SPACE SYTEMS

After a positive conclusion on availability and adequacy of the above technologies for stabilised gondolas C&C, the next question that comes in mind is of course : in all that, what can be used for other space vehicles, such as satellites?

Beyond the study of applicability in space environment, answering this question leads to analyse impact on design, development, communication and operation. That is discussed below for each of these features.

4.1 PCS AND LINUX ON BOARD : THE END OF CROSS DEVELOPMENT

PC104-based computers can easily be used on board, it depends less upon the application domain than upon the availability of a minimum amount of resources :

- electrical power consumption : few Watts under 5V DC[7], when using low-power CPU boards,
- volume and size : around 1 litre, 10×10×10 cm, without box,
- mass : around 1 kg (100gr per board + wiring), without box.

Industrial PCs' prices are extremely low (200 – 500 € per board) compared to traditional spatial computers, so that assembly and test may be more expensive than the basic components.

Mechanical / thermal environment : this kind of hardware is designed to bear industrial / factory constraints (heat, cold, dust, shock, vibration), so that no disastrous results are to be expected of space environment tests. For extremely severe environments, additional devices such as ice resistant enclosures, shock absorbing mounts, heat drains, are available off-the-shelf or can be specifically designed at minimal cost.

Radiation environment : no particular precaution is taken in industrial PCs to prevent radiation effects, so that they are probably as radiation sensitive as any industrial electronic component. Nevertheless, different solutions can be figured out (and have to be studied) to harden PC104 based computers, such as :

[7] these characteristics are of course highly dependant of the exact computer configuration : the given numbers apply to a "mean computer"

- redesigning boards on the same architecture, replacing sensitive components by hardened ones – that has been realised for some space equipment such as star trackers, and although it is a much more expensive solution compared to standard boards, it is probably cheaper than developing a specific computer;
- shielding a standard PC : why not invest saved mass and volume into a thick shielding enclosure? All radiation effect measurements are given behind 1 mm of aluminium : what does it become under 1 cm? This is the standard answer of PC104 manufacturers for extremely severe mechanical and thermal environment constraints, so it would prove relevant of the same logic.

Linux : as soon as a convenient processor is chosen, it can be used on board as well as any standard O.S. Linux is not only available for x86 targets, but also for PowerPC, ARM, MIPS, 68K, ColdFire, and some optimised kernels have been released for big 32-bit microcontrollers (100Kb RAM, 100-500Kb flash memory, network interface). On the other hand, it is still (and will stay) unavailable for small microcontrollers (8 or 16-bit, some hundred bytes RAM). On the application point of view, it is well suited for "soft real-time" applications, such as satellite platform flight software, and has proved a top level software development platform, it provides a huge amount of development tools, is well known for it is open, easy to use, flexible, powerful, reliable, and supported by the free software community (which is not the least advantage). Moreover, flight software can take advantage of numerous O.S. services to implement functions that are usually implemented from scratch in the application.

Providing an appropriate solution is found to the radiation concern[8], industrial PCs running Linux, are a convenient, standard, powerful and low-cost computer platforms for satellite systems. Since they bring on board the same development environment and techniques than those of ground software, thus virtually abolishing cross-development, they will dramatically reduces on board computer and flight software development costs, delays and difficulties.

4.2 ADA AND REAL-TIME DISTRIBUTED APPLICATIONS : C&C AT SYSTEM SCALE

Ada has been designed in the aim of real-time and embedded software, so it is not surprising that it reveals as a good solution for space flight software : it is already THE language of high-reliability software, such as civil transport and military applications. Ada compilers are available for almost every modern standard 32-bit processor (it is still virtually impossible to host Ada runtime executive on 8- or 16-bit microcontrollers), and in a variety of implementations, both on standard O.S. and on bare processors.

Of course, Ada is already used in space flight software, but only as super Pascal, thus prohibiting, for determinism and performance reasons, usage of tasking, exceptions, dynamic allocation, and even sometimes of object-oriented features and generic units. So the next step in space flight software is to break such mental blocks, and to rely on Ada high level mechanisms as well as one can rely on standard hardware and O.S.

Ada philosophy is for the programmer to put as much semantics as possible in the program, take advantage of language-inherent controls, and leave all implementation constraints to the compiler's care: that contributes to a software architecture much closer to system requirements (see 3.1) and makes mostly portable and system-independent software. Object features and generic units encourages development of reusable software and make it really possible. Combined to a design approach such as shown on Figure 2-3, it will be possible to build big portions of flight software by assembling reusable components, what will considerably ease development of on board application for families of space systems.

Annex E features allowing to develop a distributed application just as if it was a monolithic one, it is no more necessary waiting for completion of the on board computer design to start software

[8] NASA has already found some : PC104 computers fly in science payloads, particularly on the Space Shuttle

development : software application will be distributed in a second step, with criteria loosely constrained by hardware topography, and the programmer will write no communication software for that purpose. Changes in distribution configuration are so easy that software will adapt immediately to hardware evolutions. Moreover, every equipment becomes accessible, through the associated software equipment driver, from every computer : there is virtually no more separation between computers and the application runs on a "compound computer".

IP-based telecoms permit to extend Ada distribution on a system-wide scale. There will be no more distinction between a flight software and ground control centre, there will be only one control & command software running on a distributed system spreading from ground to satellite – and why not several satellites, several GCCs. Function repartition trade-off between board and ground segment will no more be a precondition for control & command developments, it will become a system integration activity, driven by data and control flow coupling concerns, less subjective than present repartition habits. Even the traditional trade distinction – flight software, ground segment – will disappear as both merge into one.

4.3 SYSTEM-WIDE NETWORK APPLICATIONS : OPERATING SATELLITES FROM YOUR DESKTOP

For balloon systems, IP-based telecoms will become very soon a reality, since a first version of the operational product will be qualified by end of 2005. It comprises a flight-compatible router, built of PC104 boards under Linux, equipped with interfaces to a dedicated L-band transmission system and to satellite telecom systems such as Inmarsat, and all usual IPV4-based network services (including QoS). New services, such as mobility or advanced security, will be added further with the help of IPV6 enhancements.

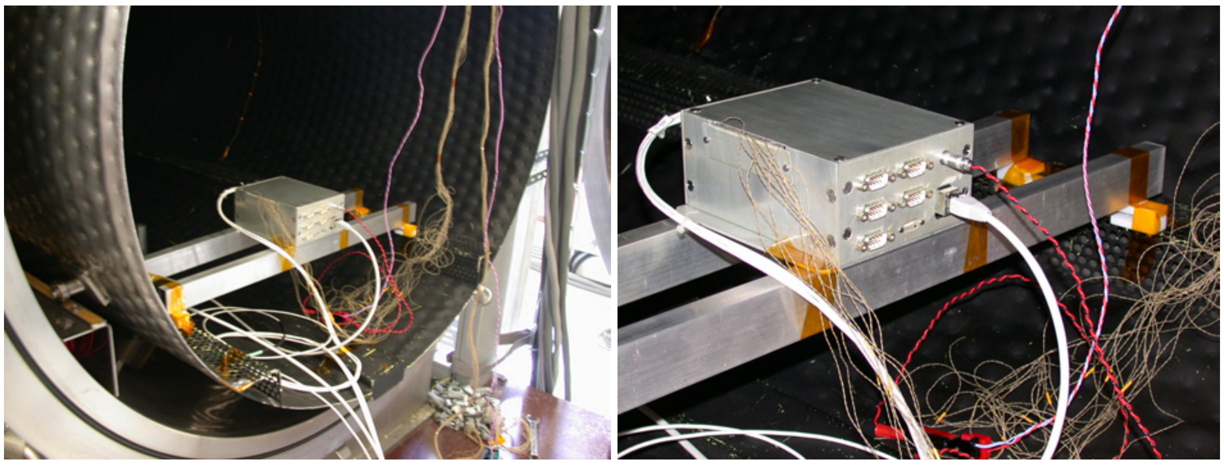

Figure 4-1 : Operational IP router after vacuum & thermal environment tests

Adoption of IP-based telecom systems for space programs, in conjunction with extensive usage of network techniques, will open new communication modes between the different components of a space system.

First, all Internet protocols and services are available in the same way for in-flight lifetime than during integration and test phases, and their specific properties may be useful for number of operational board / ground communication, e.g. :

- UDP / TCP sockets for direct traditional TM/TC flows,
- *ftp* for uploading parameter files and new flight software versions, or for downloading previously recorded telemetry,
- *http* for on-demand quick-look of on-board updated parameters,
- *telnet*, *ssh*, or equivalent, for direct communication in maintenance or configuration activities,

- *mail* for asynchronous message exchange, such as uploading or starting work plans.

IP data flows can be routed through different communication channels with homogeneous interface, whatever channel is used, so that there is no protocol difference – only data rate or persistence of communication – whether a flow is routed through a channel or another. That will allow satellite operations to resort to a series of scenarii for communication organisation and usage, such as the following.

For the needs of operating a low-orbit satellite, a permanent low-rate link established through a geostationary satellite telecom service is used for housekeeping purpose, work plan uploading and transmission of payload data quick looks. A non-permanent high-rate link established by means of a dedicated radio frequency network, is used for massive downloads of payload data. In general case, payload data download is achieved by continuous *ftp* transfers routed through the high-rate link. Depending on satellite visibility, these transfers are suspended and resumed automatically, thanks to *ftp* mechanisms. Despite the low-rate link is permanent, it may not be used continuously : GCC teams connects on the satellite periodically for routine operation, the satellite itself may send alerts upon failure and take the initiative of sending debug telemetry

As an alternative, several low-rate links may be established, for organisation reasons, e.g. separation of platform and payload flows, or separation of operation team and client access. In short, an infinity of combinations can be imagined.

Finally, since IP-based telecom uses standard protocols, supported by off-the-shelf tools available on any computer, it will be possible to operate a satellite from desktop : a client achieves programming of payload work sessions from his office, a platform chief operator connects every morning to on-board http server to check satellite health status. Thanks to mobility and extended security, operating a satellite is possible, for authorized persons, from virtually anywhere, through the web.

---

5 BIBLIOGRAPHY

[ADA ISO] JTC1/SC22/WG9 : Ada Standards – http://wwwold.dkuug.dk/JTC1/SC22/WG9/

[ADASOCK] AdaSockets – Samuel Tardieu – http://www.rfc1149.net/devel/adasockets

[CNESBAL] CNES Balloons website – http://ballons.cnes.fr/

[EAST] EAST : langage de description de données et outils associés – http://east.cnes.fr/

[GLADE] GNAT Library for Ada Distributed Execution - http://www.gnat.com/texts/products/prod_glade.htm/

[GLADE-2] GLADE, Gtk+ User Interface Builder – http://glade.gnome.org/

[GNAT] GNAT – GNU Ada Compiler – http://www.gnu.org/software/gnat/gnat.html

[GTKADA] GtkAda, a complete Ada95 graphical toolkit – http://libre.act-europe.fr/GtkAda/

[HOOD] HOOD method home page – http://www.hood-method.org/

[IP-JA00] The French Balloon Program Advancements in CNES Balloon Flight Train and Gondola Developments – André VARGAS, Jean EVRARD, Jean AUDOUBERT, CNES – *in* CO.S.PAR 33, Warsaw, Poland, July 2000.

[IP-CC00] A Command & Control Architecture organized over TCP/IP – Christine CORNIER, CNES – *in* 3rd CCSDS P1K Spacecraft On-board Interface sub-panel meeting, France, October 2000

[IP-JACC01]	The Future Command & Control Architecture for Balloons over a TCP/IP Interconnection – Christine CORNIER, Jean AUDOUBERT, CNES – *in* 15th ESA Symposium on European Rocket and Balloon Programmes and Related Research, Biarritz, France, May 2001
[LINUX]	Linux Online – http://www.linux.org/
[S-AC04]	Contrôle - commande pour les Nacelles Pointées à base de technologies standard : *Contribution à la conception et au développement du logiciel de vol* – Arnaud CHICHER, INSA 5e année AEI-TRS1, juin 2004
[S-ALDS03]	Contrôle Commande de nacelles pointées à base de technologies standard : *Rapport de Stage* – Adrien LAFONT DE SENTENAC, UPS IUP Ingénierie des Systèmes Informatiques, septembre 2003
[S-FP03]	Contrôle Commande de nacelles pointées à base de technologies standard : *Rapport de Stage* – Frédéric PORTES, UPS DESS Ingénierie des Systèmes Informatiques, juin 2003
[S-KH04]	Contrôle - commande pour les Nacelles Pointées à base de technologies standard : *Prototypage d'un segment sol* – Karim HALIOUI, EMSE ISMEA option Conception de Systèmes Informatiques, septembre 2004
[S-JPSV02]	Contrôle – commande pour les Nacelles Pointées à base de technologies standard : *Une contribution : prototypage d'applications embarquées réparties* – Jean-Pierre SEUMA-VIDAL, INSA 5e année G2I-TRS2, juin 2002
[S-SS01]	Linux embarqué pour Nacelles Pointées : *Rapport de Stage* – Semra SARPDAG, UPS IUP Ingénierie des Systèmes Informatiques, septembre 2001
[STABGND]	CNES Stabilised Gondolas website – http://ballons.cnes.fr/nacelles_pointees/
[TNI]	TNI-Europe – Object Oriented Design and Analysis Tools – http://www.tni-world.com/
[XML]	eXtensible Markup Language (XML) – http://www.w3.org/XML/
[XMLADA]	XML/Ada, a full XML suite – http://libre.act-europe.fr/xmlada/

SOMMAIRE/*SUMMARY*

Computing and control